VNU. JOURNAL OF SCIENCE, Nat. Sci., t.XV, n^{0}5 - 1999

FUNCTIONAL DEPENDENCIES WITH CONTEXT DEPENDENT NULL VALUES IN RELATIONAL DATABASES

Bui Thi Thuy Hien
Faculty of Mathematics, Mechanics and Informatics
College of Natural Sciences - VNU
Nguyen Cat Ho
Institute of Information Technology
National Center for Science and Technology.

Abstract. *The aim of this paper is to present an extension of the concept of functional dependency in a database in which the presence of context dependent null values is allowed. It is shown that the set of Armstrong's inference rules forms a sound and complete axiom system for functional dependencies under a suitable semantic of context nulls. Some rules and algorithms for manipulating context null values are also introduced and examined.*

1. INTRODUCTION

In the theory of relational database design, the integrity constraints play a crucial role and have been deeply investigated in the framework of database relations without null values. In such a framework, *functional dependencies* (FDs) are the most natural and useful. The notion of a key (derived from a given set of FDs) is fundamental to the relational model. A sound and complete axiom system for FDs was firstly given in [1] and is known as Armstrong's axiom system. Many authors, e.g. M. Levene [11], Lien[12], Atzeni and Morfuni [2], [3], [4], Maier [13] have considered FDs in database relations containing unmarked null values, which are semantically interpreted as either "unknown" [11], [13] or "no information" [12], [2], [3], [4]. Lien, Atzeni and Morfuni have introduced a sound and complete axiom system for FDs by dropping the transitivity rule and adding the union and decomposition rules to Armstrong's axiom system. To maintain the satisfaction of FDs in relational databases with incomplete information, Maier [13] has introduced and investigated marked null values.

The aim of this paper is to present an extension of the concept of functional dependency in a database in which the presence of context dependent null values is allowed. It is shown that the set of Armstrong's inference rules forms a sound and complete axiom system for functional dependencies under a suitable semantic of context null. Some rules and algorithms to manipulate context null values are also introduced and examined.

2. BASIC DEFINITIONS

Let $R(A_1, \ldots, A_n)$ be a relational scheme defined over a set of attributes $A_1, \ldots, A_n$. The domain of each attribute A_i is denoted by $Dom(A_i)$. The domain of R consists of the Cartersian product $Dom(A_1) \times Dom(A_2) \times \ldots \times Dom(A_n)$ and denoted by $Dom(R)$.

We extend each domain $Dom(A_i)$ to an extended domain $Dom^*(A_i)$ by adding a finite set of null symbols, namely $Dom^*(A_i) = Dom(A_i) \cup \Delta_{i_1} \cup \Delta_{i_2} \cup \{dne\}$, where

- Δ_{i_1} is the set of unknown context nulls denoted by $\delta_1, \delta_2, \ldots$
- Δ_{i_2} is the set of open context nulls denoted by $\beta_1, \beta_2, \ldots$
- *dne* means it "does not exit" context null.
- $Dom(A_i), \Delta_{i_1}, \Delta_{i_2}, \{dne\}$ are the disjoint sets.

The extended domain $Dom^*(R)$ of R consists of the Cartersian product

$$Dom^*(A_1) \times \ldots \times Dom^*(A_n).$$

A relation of a scheme R is a subset of $Dom^*(R)$. Such instances are denoted by lower case letters such as $r, r_1, \ldots$

A relation contains no or some null values is called a *partial relation*. The set of all partial relations over scheme R is denoted by $Rel_\uparrow(R)$.

A relation without null values called a *total relation*, the set of all total relations over scheme R is denoted by $Rel(R)$. A tuple of an instance r is called an element of r. We denote tuples by letters such as $t, t', s, s', \ldots$ If t is a tuple of a relation r, then $t[A_i]$ denotes the component of t which corresponds to the attribute A_i. If $t[A_i]$ is not null, we write $t[A_i]!$.

We use the notation *open* to refer an open context null, and notation *unk* to refer an unknown context null.

An unknown or open context null is called an indefinite value and a non null value or a *dne* null is called a definite value.

3. FUNCTIONAL DEPENDENCIES WITH CONTEXT NULL

In the classical theory, a *functional dependency* (FD) is a statement $f : X \to Y$, where X, Y are sets of attributes. A relation r over a scheme $R(U)$ (with $XY \subseteq U$) satisfies f (we say also that f holds in r) if for each pair of tuples $t_1, t_2 \in r$ such that $t_1[X] = t_2[X]$, we have $t_1[Y] = t_2[Y]$.

Let t_1 and t_2 be two tuples (may contains context nulls) over scheme R and A be an attribute. We shall write $t_1[A] == t_2[A]$) if

1. $t_1[A]!, t_2[A]!$ and $t_1[A] = t_2[A]$, or
2. $t_1[A] = \delta_i, t_2[A] = \delta_j$ and $i = j$, or
3. $t_1[A] = \beta_j, t_2[A] = \beta_i$ and $i = j$, or
4. $t_1[A] = dne$ and $t_2[A] = dne$.

By $t_1[X] == t_2[X]$ we understand $t_1[A] == t_2[A]$, for all $A \in X$.

Therefore, the meaning of comparison operator $==$ is just to check the symbolic equality of values in the database. For instance $3 == 3; \delta_i == \delta_i; \beta_i == \beta_i$ and $dne == dne$.

Contrary to comparison operator $==$ is the comparison operator $=/=$. For example $3 =/= 4; \delta_i =/= \delta_j; \delta_i =/= \beta_i$.

In databases with incomplete information, a natural question arises at this juncture is that: what is the truth value of $x = y$ if x or y or both are null? In [9] Codd has introduced a three-valued logic $\{0, \omega, 1\}$ for using to exploit data from databases that may contain null values. In the case of context nulls, we adopt a five-valued logic to compare the values in a context null database. The following table shows us the truth value assignment to such comparison:

$=$	a	b	δ_i	δ_j	β_i	β_j	dne
a	1	0	ω	ω	ω	ω	0
b	0	1	ω	ω	ω	ω	0
δ_i	ω	ω	ξ	ξ^-	ξ^-	ξ^-	0
δ_j	ω	ω	ξ^-	ξ	ξ^-	ξ^-	0
β_i	ω	ω	ξ^-	ξ^-	ξ	ξ^-	ξ^-
β_j	ω	ω	ξ^-	ξ^-	ξ^-	ξ	ξ^-
dne	0	0	0	0	ξ^-	ξ^-	1

Fig. 1.

If we define v a truth value assignment to a comparison between any two values in a context null database, then we have $v(a = b) = 0; v(a = dne) = 0; v(\delta_i = \delta_j) = \xi^-; v(\delta_i = \delta_i) = \xi; v(a = a) = 1; v(dne = dne) = 1; \ldots$

We now extend the concept of functional dependency in a context null database.

Definition 1. Let $R(U)$ be a relation scheme; $X, Y \subseteq U$; ε be a truth value. $f : X \overset{\varepsilon}{\rightarrow} Y$ *is said to be a functional dependency with context nulls over* $R(U)$ (cnFD) if for each context null relation r over $R(U)$ and for each pair of tuples $t_1, t_2 \in r$ such that $v(t_1[X] = t_2[X]) \geq \varepsilon$, we have $v(t_1[Y] = t_2[Y]) \geq \varepsilon$. If $\varepsilon = \xi$, we write $X \rightarrow Y$ instead of $X \overset{\varepsilon}{\rightarrow} Y$.

Proposition 1. *$v(t_1[X] = t_2[X]) \geq \xi$ if and only if $t_1[X] == t_2[X]$.*

Proof. The proof is directly deduced from the definition of function v and comparison operator $==$.◇

From Proposition 1, it follows that Definition 1 is equivalent to the following one:

Definition 2. Let $R(U)$ be a relation scheme and $X, Y \subseteq U$; $f : X \rightarrow Y$ *is said to be a functional dependency with context nulls* over $R(U)$ (cnFD) if for each context null relation r over $R(U)$ and for each pair of tuples $t_1, t_2 \in r$ such that $t_1[X] == t_2[X]$, we have $t_1[Y] == t_2[Y]$.

In this paper, we restrict ourselves to investigate functional dependencies under context nulls $X \overset{\varepsilon}{\rightarrow} Y$ with $\varepsilon = \xi$, i.e., cnFDs of the form $X \rightarrow Y$.

r	A	B	C
	δ_1	β_1	2
	δ_1	β_1	3

Fig. 2.

Example 1. The relation r given in Fig. 2 satisfies functional dependency $A \to B$.

Let F be a set of data dependencies, a data dependence f is called a logical consequence of F if any relation r satisfies F then it also satisfies f.

We denote $F^* = \{f \mid f \text{ is a logical consequence of } F\}$.

Given a set of inference rules, set $F^+ = \{f \mid f \text{ is deduced from } F \text{ by means of the inference rules } \}$.

The set of inference rules is said to be sound if $(F^+ \subseteq F^*)$ and to be complete if $(F^+ = F^*)$.

It is well known that for functional dependencies in the relational model without nulls, the following is a sound and complete set of inference rules:

A_1) reflexivity: If $Y \subseteq X$ then $X \to Y$

A_2) augmentation: If $X \to Y$ then $XZ \to YZ$

A_3) transitivity: If $X \to Y$ and $Y \to Z$ then $X \to Z$.

From rules A_1, A_2, A_3 we can deduce the following two rules:

A_4) union: If $X \to Y$ and $X \to Z$ then $X \to YZ$

A_5) decomposition: If $X \to YZ$ then $X \to Y$.

According to Definition 1 (or Definition 2) it is easy to see that reflexivity, augmentation, transitivity, union and decomposition rules are sound also for functional dependencies with context nulls.

For convenience, we recall here the following notion: The closure X^+ of a set of attributes X with respect to a set F of functional dependencies with context nulls is defined as follows:

$X^+ = \{A \mid X \to A \text{ is deduced from } F \text{ by means of the inference rules } A_1, A_2, A_3\}$

By the rules of union and decomposition, it is clear that $X \to Y$ is deduced from F by means of the rules if and only if $Y \subseteq X^+$.

Theorem 1. *The rules A_1, A_2, A_3 form a sound and complete set of inference rules for functional dependencies with context nulls.*

Proof. a) Soundness of these rules has been shown above.

b) Completeness: Let F be a set of functional dependencies with context nulls over a scheme $R(U)$, $F^* = \{f \mid f \text{ is logical consequence of } F\}$, $F^+ = \{f \mid f \text{ is deduced from } F \text{ by means of } A_1, A_2, A_3\}$. Since A_1, A_2, A_3 are sound, we have $F^+ \subseteq F^*$. It remains to prove that $F^* \subseteq F^+$, that means we need to show: if $f \in F^*$ then $f \in F^+$. This is equivalent to show that if $f \notin F^+$ then $f \notin F^*$.

Assume that $g : X \to Y$ is a cnFD and $g \notin F^+$. Let r be a two tuple relation $\{t_1, t_2\}$, where

$\forall A_i \in U$, then $t_1[A_i] = 1$ or

$t_1[A_i] = \delta_i$ or

$t_1[A_i] = \beta$, or
$t_1[A_i] = dne$,
$\forall A_i \in X^+$ then $t_2[A_i] == t_1[A_i]$
$\forall A_i \in U \setminus X^+$ then $t_2[A_i] = / = t_1[A_i]$

(1) r satisfies all the dependencies in F: Let $(V \to W) \in F$, assume $V \to W$ is not satisfied by r, i.e., $t_1[V] == t_2[V]$ and $t_1[W] = / = t_2[W]$. From $t_1[V] == t_2[V]$ we have $V \subseteq X^+$. From $t_1[W] = / = t_2[W]$ it implies that W must contain at least one attribute in $U \setminus X^+$, say A. Thus, $\exists A \in W$ such that $A \notin X^+(*)$. Since $V \subseteq X^+$ we conclude that $(X \to V) \in F^+$. Indeed, from $(V \to W) \in F$, it follows by A_3 that $(X \to W) \in F^+$, and so $W \subseteq X^+$. But, it is impossible because of (*). Hence, r satisfies all the dependencies in F.

(2) r does not satisfy g : Assume the contrary that r satisfies $g : X \to Y$. From $X \subseteq X^+$ we deduce $t_1[X] == t_2[X]$ and hence, $X \to Y$. Thus, we have $t_1[Y] == t_2[Y]$. It implies by definition of t_1, t_2 that $Y \subseteq X^+$. So, $(X \to Y) \in F^+$, a contradiction.

Therefore, $F^* \subseteq F^+$. Combining with $F^+ \subseteq F^*$ we have $F^* = F^+$.◇

4. SOME RULES TO MAINTAIN THE SATISFACTION OF FUNCTIONAL DEPENDENCIES IN CONTEXT NULL DATABASES

According to the semantic approach to context nulls, context null values is defined by well-known information. The set of functional dependencies are, of course, very important well-known information to define context nulls. That means, context nulls have to be defined and handled to ensure that the database with context nulls under consideration still satisfies a given set of functional dependencies. Hence, while implementing the data update procedures, the system has to maintain the satisfaction of functional dependencies in the database. To obtain this objective, some rules for handling context nulls need to be obeyed.

Definition 3. Let r be in $Rel_\uparrow(R)$, $X \to A$ be a cnFD over R, t_1 and t_2 be two tuples of r such that $t_1[X] == t_2[X]$,

- If $t_1[A]!, t_2[A]!$ and $t_1[A] = / = t_2[A]$ then r has a hard violation of $X \to A$ and, t_1 and t_2 are said to cause a hard violation of $X \to A$,
- If $t_1[A] = / = t_2[A]$ (*) and one side of (*) is *dne* and the other side is not an open context null then r has a hard violation of $X \to A$.
- If $t_1[A] = / = t_2[A]$ (*), t_1 and t_2 do not cause a hard violation of $X \to A$ and at least one of two side of (*) is null then r has a soft violation of $X \to A$ and, t_1 and t_2 are said to cause a soft violation of $X \to A$.

Example 2. Let $X \to A$ is a cnFD over R and consider two tuples t_1 and t_2 of r and suppose that $t_1[X] == t_2[X]$, then,

- if $t_1[A] = 1, t_2[A] = 3$ then r has a hard violation of $X \to A$
- if $t_1[A] = dne$, $t_2[A] = \delta_i$ then r has a hard violation of $X \to A$
- if $t_1[A] = dne$, $t_2[A] = 1$ then r has a hard violation of $X \to A$

- if $t_1[A] = dne, t_2[A] = \beta_i$ then r has a soft violation of $X \to A$

The function VIOLATION in Algorithm 1 will check whether two tuples t_1 and t_2 cause a violation of the cnFD $X \to A$.

Algorithm 1. $VIOLATION(r, t_1, t_2, X \to A)$

Input : $r \in Rel_\uparrow(R), X \to A$ is a cnFD over R, t_1 and t_2 are any two tuples of r such that $t_1[X] == t_2[X]$.

Output : 2 if t_1 and t_2 cause a hard violation of $X \to A$; 1 if t_1 and t_2 cause a soft violation of F; 0 otherwise.

```
Begin
$VIOLATION := 0$;
if ($t_1[A]!$ and $t_2[A]!$ and $t_1[A] = / = t_2[A]$) then $VIOLATION := 2$
    else if ($t_1[A] = / = t_2[A]$) and (($t_1[A] == dne$
        and ($t_2[A] = / = open$) or (( $t_2[A] == dne$) and ($t_1[A] = / = open$))
        then $VIOLATION := 2$;
        else if ($t_1[A] = / = t_2[A]$) then $VIOLATION := 1$;
End.
```

Definition 4. A context null database is said to be consistent with a given set F of functional dependencies if there is not any hard or soft violation of F in the database.

Definition 5. Let DB be a context null database and r be a relation in DB and F be a set of cnFDs over R. Assume that t_1 and t_2 are any two tuples in r that cause a soft violation of an $X \to A$ in F. A soft violation removal that is caused by t_1 and t_2 in r (or in DB), is defined as follows:

1. If one of the two values $t_1[A]$ and $t_2[A]$ is not null, (say $t_1[A]$) and the other ($t_2[A]$) is either an unknown context null or an open context null, then every occurrence of the null value $t_2[A]$ in r (or in DB) is changed by $t_1[A]$.

2. If one of the two values $t_1[A]$ and $t_2[A]$ is either a dne null or an unknown context null (say $t_1[A]$) and the other value ($t_2[A]$) is an open context null, then every occurrence of the open null ($t_2[A]$) in r (or in DB) is changed by $t_1[A]$.

3. If both $t_1[A]$ and $t_2[A]$ are either unknown context nulls or open context nulls, then every occurrence of the one with greater index (say $t_2[A]$) in r (or in DB) is changed by the other with smaller index ($t_1[A]$).

Example 3. In the cases below, the soft violations in r (or DB) will be removed as follows:

If $t_1[A]!$ and $t_2[A] = \delta_i$ then every occurrence of δ_i in r (or in DB) is changed by $t_1[A]$.

If $t_2[A]!$ and $t_1[A] = \beta_i$ then every occurrence of β_i in r (or in DB) is changed by $t_2[A]$.

If $t_1[A] = dne$ and $t_2[A] = \beta_i$ then every occurrence of β_i in r (or in DB) is changed by dne.

If $t_1[A] = \delta_i$ and $t_2[A] = \beta_j$ then every occurrence of β_j in r (or in DB) is changed by δ_i.

If $t_1[A] = \delta_i, t_2[A] = \delta_j$ and $i < j$ then every occurrence of δ_j in r (or in DB) is changed by δ_i.

If $t_1[A] = \beta_i, t_2[A] = \beta_j$ and $i < j$ then every occurrence of β_j in r (or in DB) is changed by β_i.

Before presenting an algorithm for removing a soft violation that appears in a relation or in a DB, let us first show an algorithm which changes every occurrence of a null value in a relation (or in a DB) to a definite value or to a more information null value. The algorithm CHANGE in Algorithm 2 changes all the value x at the column of attribute A in $M(M = r$ or $M = DB)$ to value y.

Algorithm 2. $CHANGE(x, A, M, y)$

Input : $r \in Rel_{\uparrow}(R), M$ is a context null database or M is a relation r, A is an attribute column under consideration, x and y are two values (may be null) at the attribute column A.

Output : Change every occurrence of value x at attribute column A in M to value y.

```
Begin
   For each relation $r$ in $M$ do
      For each tuple $t$ in $r$ do
         if $t[A] == x$ then $t[A] := y$;
End;
```

The algorithm for removing a soft violation in a relation or in a DB is presented as Algorithm 3.

Algorithm 3. $REMOVEVIO(M, t_1, t_2, X \to A)$

Input : $r \in Rel_{\uparrow}(R), M$ is a context null database or M is a relation $r, X \to A$ is a cnFD over R and, t_1 and t_2 are tuples which cause a soft violation of $X \to A$.

Output : Remove the soft violation caused by t_1 and t_2.

```
Begin
if  $t_1[A]$! and $t_2[A] == unk$ then
   begin $x := t_2[A]; CHANGE(x, A, M, t_1[A])$; end;
if  $t_2[A]$! and $t_1[A] == unk$ then
   begin $x := t_1[A]; CHANGE(x, A, M, t_2[A])$; end;
if  $t_1[A]$! and $t_2[A] == open$ then
   begin $x := t_2[A]; CHANGE(x, A, M, t_1[A])$; end;
if  $t_2[A]$! and $t_1[A] == open$ then
   begin $x := t_1[A]; CHANGE(x, A, M, t_2[A])$; end;
if  $t_1[A] == dne$ and $t_2[A] == open$ then
   begin $x := t_2[A]; CHANGE(x, A, M, dne)$; end;
if  $t_1[A] == unk$ and $t_2[A] == open$ then
   begin $x := t_2[A]; y := t_1[A]$; $CHANGE(x, A, M, y)$; end;
if  $t_2[A] == dne$ and $t_1[A] == open$ then
   begin $x := t_1[A]; CHANGE(x, A, M, dne)$; end;
if  $t_2[A] == unk$ and $t_1[A] == open$ then
   begin $x := t_1[A]; y := t_2[A]$; $CHANGE(x, A, M, y)$; end;
```

```
    if ($(t_1[A] == unk$ and $t_2[A] == unk)$ or $(t_1[A] == open$ and $t_2[A] == open)$ then
      begin
      $x := t_1[A]; y := t_2[A];$
      if $index(x) > index(y)$ then $CHANGE(x, A, M, y)$
      else $CHANGE(y, A, M, x)$
      end;
    End.
```

Lemma 1. *Let r be in $Rel_\uparrow(R), F$ be a set of cnFDs over R. If two tuples t_1 and t_2 of r cause a soft violation of F and r' is the relation deduced from r by moving the soft violation caused by t_1 and t_2, then $r' \geq r.\diamondsuit$*

Proof. By Definition 5, if a soft violation is removed then:

(i) The definite values in r are unchanged.

(ii) Each null value in r is either unchanged or changed to a definite value or changed to a more information null value.

Combining (i) and (ii) we have $r' \geq r.\diamondsuit$

Theorem 2. *Let DB be a context null database, r be a relation in DB and F be a set of cnFDs over R. If the following conditions hold:*

1. In DB, there is not any hard violation of F.

2. All the soft violations of F that appears in DB can be removed such that in DB there is not any hart violation of F,

then $DB_{old} \subseteq DB_{new}$.

Proof. Directly deduced from Lemma 1.$\diamondsuit$

Lemma 2. *Let r be in $Rel_\uparrow(R), F$ be a set of cnFDs over R, t be a tuple over R. If the following conditions hold:*

- in r there is not any hard violation of F,

- between t and r there is not any hard violation of F,

- between t and r there is a soft violation of F,

then after removing this soft violation, we have:

(i) in r there is not any hard violation of F,

(ii) between t and r there is not any hard violation of F.

Proof. Suppose t' is a tuple of r, t and t' cause a soft violation of $X \to A$ in F. When removing a soft violation between t and t' , there are two the following possibilities:

Case 1: The tuple t is to be changed at the value $t[A]$ and the tuple t' is kept unchanged:

In this case, the relation r is not changed. By the first condition of the assumption we have (i).

To prove (ii), we suppose the contrary, that there is a tuple t_1 of r such that t and t_1 cause a hard violation (HV) of F. Since t is only be changed at value $t[A]$ and in the initial relation r there is not any HV, so if t and t_1 cause a HV of F then such HV must be HV of cnFD $X \to A$. Since after removing the soft violation (SV) between t and t' we have $t[A] == t'[A]; t[X] == t'[X]$, if t and t_1 cause a HV of cnFD $X \rightarrow A$. then t' and t_1 also must cause a HV of cnFD $X \rightarrow A$. This contradicts (i).

Case 2: The tuple t' is changed at value $t'[A]$ and the tuple t is kept unchanged:

(i): On the contrary, suppose the assertion (i) does not hold.

By the first condition of the assumption, there is not any HV in r, therefore, if a HV that appears in r after removing the SV between t and t' then such HV must be the HV between t' and a tuple t_1 of r'. Since t' is only changed at value $t'[A]$, the HV between t' and t_1 must be of cnFD $X \rightarrow A$. Since after removing the SV between t and t' we have $t[A] == t'[A]$ and $t[X] == t'[X]$, if t' and t_1 cause a HV of cnFD $X \rightarrow A$ then t and t_1 cause also a HV of cnFD $X \rightarrow A$. Clearly, t_1 is the tuple of r before removing the SV between t and t'. This contradicts assumption that t and r do not cause any HV of F.

(ii): Since the tuple t is kept unchanged, so for any $t_1 \in r'$, and if $t_1 \neq t'$, t and t_1 do not cause any HV of F. It remains to check that whether t and t' cause a HV of F or not? Suppose, t and t' cause a HV of cnFD $Y \rightarrow B$. Because after removing the SV between t and t' the tuple t' is only changed at value $t'[A]$, the initial tuples t and t' must cause also a HV of cnFD $Y \rightarrow B$. This contradicts the second condition of the assumption.$\diamondsuit$

Theorem 3. *Let r be in $Rel_{\top}(R)$, F be the set of cnFDs over R, t be a tuple over scheme R. If the following conditions are satisfied:*

- in r there is not any soft or hard violation of F,

- between t and r there is not any hard violation of F. then after removing all the soft violations appear beween t and r, we have:

(i) In r there is not any hard violation of F,

(ii) Between t and r there is not any hard violation of F,

(iii) In r there is not any soft violation of F.

Proof. (i) and (ii): Consider any tuple t' in r and suppose that t and t' cause a SV of F. By Lemma 2, after removing the SV between t and t' , we have:

- in r there is not any HV,

- between r and t there is not any HV.

Applying Lemma 2 to the tuple t and the relation r until all the SVs between t and r are removed, we obtain (i) and (ii).

(iii): Assume the contrary, that after removing all the SVs between t and r, there are a cnFD $X \rightarrow A$ in F and two tuples t' and t_1 of r such that t' and t_1 cause a SV of $X \rightarrow A$. Then, $t'[X] == t_1[X]$ and $t'[A] =/= t_1[A]$ (1). Since there is not any SV in the initial relation r, the appearance of the SV between t' and t_1 shows that at such a time either t' and t or t_1 and t do cause a SV of cnFD $X \rightarrow A$. Indeed, suppose the contrary that, both t' and t, t_1 and t does not cause any SV of cnFD $X \rightarrow A$. It shows that the values $t[A]$ and $t_1[A]$ are not changed. Since there is not any SV in the initial relation r, if $t'[X] == t_1[X]$ then $t'[A] == t_1[A]$ which contradicts (1). Therefore, at such a time either t' and t or t_1 and t must cause a SV of cnFD $X \rightarrow A$, that means, $t[X] == t'[X]$. Since all the SVs between t and r have been removed and $t[X] == t'[X]$, we get $t[A] == t'[A]$ and $t[A] == t_1[A]$. It follows that $t'[A] == t_1[A]$, a contradiction to (1).$\diamondsuit$

By the Theorem 2 and the Theorem 3, in order to maintain the consistency of a context null database, it is necessary to introduce the following two rules for updating and inserting data:

Rule 1. (*For updating data*)

Let DB be a context null database and r be a relation in DB; F be a set of cnFDs over R. Let t be a tuple of r that needs to be updated to become t_1:

(i) If there is a tuple t' in $r \setminus \{t\}$ that t_1 and t' cause a hard violation of F then the system will not implement the update procedure for the tuple t.

Conversely, if (i) is not satisfied:

(ii) For each $X \to A$ in F, the system will implement checking:

For each tuple t' in $r \setminus \{t\}$, if t_1 and t' cause a soft violation of F then the system will remove that soft violation.

When all soft violations between $r \setminus \{t\}$ and the tuple t_1 have been removed, the system will implement updating the tuple t to become the tuple t_1.

The aim of Rule 1 is to maintain the database under consideration to be consistent with a given set F of functional dependencies, i.e., Rule 1 ensures that updating procedure does not cause any violation of F. Hence, the Rule 1 is said to be correct if it realizes this aim.

Proposition 2. *The Rule 1 is correct.*

Proof. The proof of (i) is straightforward.

For (ii): Let $r' = r \setminus \{t\}$. Rule 1 shows that:

- There is not any HV and any SV in r',

- t and r does not cause any HV of F,

By application of Theorem 3 to the relation r and the tuple t, all the SVs between r and t can be all removed so that:

(i) There is not any HV of F in r'

(ii) There is not any HV of F between r and t.

(iii) There is not any SV of F in r'.

Therefore, Rule 1 is correct, by Theorem 3.◇

Algorithm 4 below uses Rule 1 to update a tuple t of relation r to become a tuple t_1:

Algorithm 4. $UPDATE(r, F, t, t_1)$

Input : $r \in Rel_{\uparrow}(R)$, a set of cnFDs F over R, a tuple t_1 over R, a tuple t of r that need to be updated to become the tuple t_1.

Output : Update the tuple t to become t_1 if there is not any HV of F that appears between $r \setminus \{t\}$ and t_1, do not update if otherwise.

```
Begin
$r' := r \setminus \{t\}$;
For each $X \to A$ in $F$ do
    For each tuple $t$ in $r'$ do
    if $t[X] == t_1[X]$ then
    if $VIOLATION(r', t, t_1, X \to A) = 2$ then exit;
  For each $X \to A$ in $F$ do
    Repeat
    mark:=0;
    For each tuple $t$ in $r'$ do
      if $t[X] == t_1[X]$ then
      if $VIOLATION(r', t, t_1, X \to A) = 1$ then
      Begin
      $REMOVEVIO(DB, t, t', X \to A)$;
      mark:=1;
      End;
    Until mark = 0;
  $r := r' \cup \{t_1\}$;
  End.
```

Example 4. Consider the relation $r(ABCD)$ given in Fig. 3 and a set of functions $F = \{A \to C, C \to D\}$

r	A	B	C	D
t_1	a_1	b_1	c_1	dne
t_2	a_2	δ_2	dne	d_1
t_3	a_1	b_2	c_1	dne

Fig.3.

Suppose that the system is required to update the tuple $t_3 = (a_1, b_2, c_1, dne)$ to become a tuple $t = (a_1, b_2, c_1, 4)$. Since $t[C] == t_1[C]$ and $t[D] == 4, t_1[D] == dne$; t and t_1 cause a HV of F in r. Therefore, the system does not update the tuple t_3 to the tuple t by the Rule 1.

Rule 2 (*For inserting data*)

Let DB be a context null database and r be a relation in DB, F be a set of cnFDs over R. Let t be a tuple that needs to be inserted into r.

(i) If there is a tuple t' in r such that t and t' cause a hard violation of F, then the system does not implement the insert procedure for the tuple t.

Conversely, if (i) is not satisfied, then

(ii) For each $X \to A$ in F, the system will implement checking:

For each tuple t in r, if t and t' cause a soft violation of F then the system will remove that soft violation.

Whenever all soft violation between r and the tuple t have been removed, then the system will implement inserting the tuple t into relation r.

The aim of Rule 2 is to maintain the database under consideration to be consistent with a given set F of functions, i.e., Rule 2 ensures that inserting procedure does not cause any violation of F. Hence, the Rule 2 is said to be correct if it realizes this aim.

Proposition 3. *The Rule 2 is correct.*

Proof. The proof is similar to that of Proposition 2.◇

Algorithm 5 uses Rule 2 to insert the tuple t into relation r.

Algorithm 5. $INSERT(r, F, t)$

Input : $r \in Rel_{\uparrow}(R)$, a set of cnFDs F over R, a tuple t which needs to be inserted into relation r.

Output : Insert the tuple t into r if there is not any HV of F that appears between r and t, do not insert if otherwise.

```
Begin
For each $X \to A$ in $F$ do
For each tuple $t'$ in $r$ do
    if $t'[X] == t[X]$ then
        if $VIOLATION(r, t, t', X \to A) = 2$ then exit;
For each $X \to A$ in $F$ do
    Repeat
    mark:=0;
    For each tuple $t'$ in $r$ do
        if $t'[X] == t[X]$ then
            if $VIOLATION(r, t, t', X \to A) = 1$ then
            Begin
            $REMOVEVIO(DB, t, t', X \to A)$;
            mark:=1;
            End;
    Until mark = 0;
$r := r \cup \{t\}$;
End.
```

Example 5. Consider a relation $r(ABCD)$ given in Fig. 4 and a set of functions $F = \{A \to C, C \to D\}$

r	A	B	C	D
t_1	a_1	b_1	δ_1	β_1
t_2	a_2	δ_2	dne	d_1

Fig.4.

Assume that a tuple $t = (a_1, b_2, dne, d_2)$ needs to be inserted into relation r. Since $t_1[A] == t[A]$ but $t[C] == dne$ and $t_1[C] == \delta_1$, by Definition 2, t and t_1 cause a HV of $A \to C$ in r. So the tuple t do not be inserted into r by the Rule 2.

Assume a tuple $t = (a_1, b_2, c_1, dne)$ needs to be inserted into relation r. Since $t_1[A] == t[A]$ but $t[C] == c_1$ and $t_1[C] == \delta_1$, by the Definition 2, t and t_1 cause a

r	A	B	C	D
t_1	a_1	b_1	c_1	dne
t_2	a_2	δ_2	dne	d_1
t_3	a_1	b_2	c_1	dne

Fig 5.

SV of $A \to C$ in r. By Rule 2, the system will remove the above soft violation by changing $t_1[C]$ to c_1. When changing $t_1[C]$ to c_1, since $t_1[C] == t[C]$ and $t[D] == dne$ and $t_1[D] == \beta_1$, the tuples t and t_1 cause a soft violation of $C \to D$ in r. This SV will be removed by changing $t_1[D]$ to dne. Then there is no any HV and SV of F that appears in r. By Rule 2, the tuple t will be inserted into r. The Fig. 5 presents the obtained results.

Theorem 4. *A context null database always is always consistent with a given set of functional dependencies if it obeys Rule 1 and Rule 2.*

Proof. As a direct consequence of Lemma 1 and Lemma 2.◇

5. CONCLUSION

In this paper we have presented an extension of the concept of functional dependency to a framework in which the presence of null values is allowed under the context dependent interpretation. Functional dependencies with context nulls have been defined and valid inference rules have been presented. It is shown that the set of Armstrong's inference rules forms a sound and complete system of axioms for functional dependencies with context nulls as well. This allows us to utilise functional dependencies as a design tool for relational schemes in presence of context dependent null values. The results in the paper show that sets of functional dependencies are also important well-known information to define the values of context null.

REFERENCES

1. W.W. Armstrong. *Dependency structures of database relationships*, Proceedings of the IFIP Congress, Stockholm, 1974, 580-583.
2. P. Atzeni and N. M. Morfuni. Functional Dependencies in Relations with Null Values, *Inform. Process. Lett.* **18**, May 1984, 233-238.
3. P. Atzeni and N. M. Morfuni. Functional Dependencies and Disjunctive Existence Constraints in database Relations with Null Values, in "11th Colloq. Automata, Lang. Programming (ICALP)", *Lecture Notes in Comput. Sci.* Vol. **172** (1984) 69-81.
4. P. Atzeni and N. M. Morfuni. *Functional Dependencies and Constraints on Null Values in Database Relations*, Academic Press, New York and London, **70** (1), July 1986, 1 - 31.

5. E. F. Codd. Extending the database relational model to capture more meaning, *ACM TODS*, **4** (4) (1979) 397-434.
6. B. S. Goldstein. Constraints on Null Values in Relational Databases, in "Proc. 7th Internat. Conf. on Very Large Data Bases", IEEE Computer Society, Los Angeles, 1981, 101-110.
7. Bui Thi Thuy Hien. Relational databases with context dependent null values, *Journal of Computer Science and Cybernetics* **15** (1) (1999) 20 - 30.
8. Nguyen Cat Ho, Le The Thang. The semantic of data in databases with incomplete information, *Journal of Computer Science and Cybernetics* **11** (2) (1995) 7-15.
9. Nguyen Cat Ho. A relational model of databases with context dependent null values, *Bull. Pol. Ac. Tech.*, Vol. **36** (1-2) (1988) 77-90.
10. Nguyen Cat Ho. Context Dependent Null Values and Multivalued Dependencies in Relational Databases. *Bull. Pol. Ac. Tech.*, Vol. **36** (1-2) (1988) 91-105.
11. M. Levene and G. Loizou. The additivity problem for functional dependencies in incomplete relations, *Acta Informatica* **34** (1997) 135-149.
12. Y. E. Lien. On the Equivalence of Database Models, *Journal of the ACM* **29** (2), April 1982, 333-362.
13. D. Maier. *The Theory of Relational Databases*, Computer Science Press, 1983.

TẠP CHÍ KHOA HỌC ĐHQGHN, KHTN, t.XV, n^{0}5 - 1999

PHỤ THUỘC HÀM VỚI NULL NGỮ CẢNH TRONG CƠ SỞ DỮ LIỆU QUAN HỆ

Bùi Thị Thúy Hiền
Khoa Toán - Cơ- Tin học - Đại học KH Tự nhiên - ĐHQG Hà Nội
Nguyễn Cát Hồ
Viện Công nghệ Thông tin - TTKH & CNQG

Mục đích của bài báo này là mở rộng khái niệm phụ thuộc hàm trong cơ sở dữ liệu quan hệ có null ngữ cảnh. Nó chỉ ra, dưới ngữ nghĩa của null ngữ cảnh, các qui tắc suy diễn của Armstrong tạo nên một hệ tiên đề xác đáng và đầy đủ cho phụ thuộc hàm. Một vài qui tắc và thuật toán để thao tác null ngữ cảnh cũng được giới thiệu và kiểm tra.